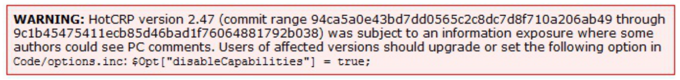
Poster: Security in Web-Based Workflows

Thomas Bauereiß[*], Abhishek Bichhawat[†], Iulia Boloşteanu[‡], Peter Faymonville[†], Bernd Finkbeiner[†], Deepak Garg[‡], Richard Gay[¶], Sergey Grebenshchikov[§], Christian Hammer[†], Dieter Hutter[*], Ondřej Kunčar[§], Peter Lammich[§], Heiko Mantel[¶], Christian Müller[§], Andrei Popescu[§,‖], Markus Rabe[†], Vineet Rajani[‡], Helmut Seidl[§], Markus Tasch[¶] and Leander Tentrup[†]
[*]German Research Center for Artificial Intelligence (DFKI), Bremen, Germany, Email: {firstname.lastname}@dfki.de
[†]Saarland University, Germany, Email: {lastname}@cs.uni-saarland.de
[‡]MPI-SWS, Saarbrücken, Germany, Email: {iulia_mb, dg, vrajani}@mpi-sws.org
[§]Technische Universität München, Germany, Email: {lastname}@in.tum.de
[¶]Technische Universität Darmstadt, Germany, Email: {lastname}@mais.informatik.tu-darmstadt.de
[‖]Dept. of Computer Science, School of Science and Technology, Middlesex University, UK, Email: a.popescu@mdx.ac.uk

I. Motivation

Modern day web consists of a variety of complex workflow applications (e.g., Amazon, Facebook, EasyChair, Microsoft HealthVault, etc.), which involve processing of sensitive data on both the server and the browser (client) side. These systems often deal with confidential information like credit card details, medical and health-related data, location information and sensitive documents. This makes them interesting targets for a variety of attacks [1]–[6].

A main characteristic of the mentioned workflow/document management systems are the *complex requirements* on their flow of information: on the one hand, information is supposed to flow quite intensively; on the other hand, certain information needs to be restricted from various parties under various circumstances. Even assuming the absence of "malicious" parties, there is an intrinsic concern about the prevention or detection of programming errors that can lead to exposure of confidential information. For example, a conference's program committee members would surely sleep better if the following type of behavior (exhibited by an older version of the HotCRP conference management system) did not occur:


WARNING: HotCRP version 2.47 (commit range 94ca5a0e43bd7dd0565c2c8dc7d8f710a206ab49 through 9c1b45475411ecb85d46bad1f76064881792b038) was subject to an information exposure where some authors could see PC comments. Users of affected versions should upgrade or set the following option in `Code/options.inc: $Opt["disableCapabilities"] = true;`

At Reliably Secure Software Systems (RS^3) [7], we develop semantically justified information-flow analysis and verification. The research landscape of information-flow system security is divided between theoretical work proposing elaborate security notions and proof methods, and practical implementations usually focusing on enforcing simpler, tractable properties. As part of the RS^3 *reference scenarios*, we aim to bridge this gap between theory and practice. Here we present our work on one of the three reference scenarios (whose name gives this abstract's title). We outline the techniques we develop to a large extent independently within different projects and our efforts to integrate these techniques into a realistic end product: CoCon, a conference management system with formally verified confidentiality guarantees. CoCon was successfully used at a workshop [8] and is currently evaluated for being used at a full-fledged conference [9]. Along with CoCon, we propose a reusable methodology for the holistic (server and client) verification of web-based systems. We highlight the need for server-client communication to prevent illicit information flows on the client side, which might otherwise leak sensitive data.

II. Approach

A typical web application broadly consists of two parts: one that runs on the server and one that executes in a web browser. Our approach to achieve holistic security guarantees is the following (Figure 1):

A We develop the application logic of the server part (a RESTful API) inside a theorem prover, mechanically verify its information-flow security properties, and use the code extraction feature of the theorem prover to obtain executable code with the same properties.

B On the client side, a runtime monitor in the browser, covering the JavaScript (JS) interpreter, the document object model (DOM) and the event handling mechanism, provably enforces a variant of reactive non-interference: secret data does not influence publicly observable data during JS code execution.

C The client-side monitor needs to be aware of which data flows are (il)legal. To this end, a security policy is transmitted from the server along with HTML content.

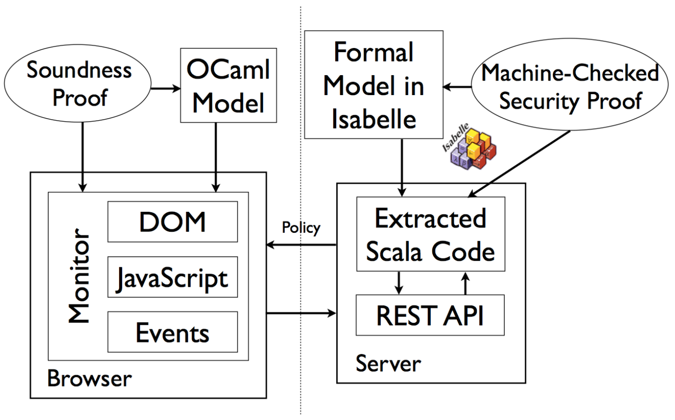


Fig. 1: Overview of the approach

III. Results and Ongoing Work

A. Server Verification

A conference management system has complex confidentiality requirements, such as: "authors learn nothing about their paper's reviewer assignment beyond the number of reviewers, and only in the notification phase." We have designed a parameterized security notion called *Bounded Deducibility (BD) security* to capture such properties [6]: its parameters are the sources, the sinks, a declassification bound ("beyond" which information should not flow) and a declassification trigger ("unless" which information should not flow)—the last two expressed as arbitrary formulas in higher-order logic (HOL). We have formalized both BD security and the core of CoCon in the Isabelle interactive theorem prover [10] and verified the desired confidentiality properties for the relevant information sources: papers, reviews, reviewer assignment, etc. Executable code in Scala was extracted automatically from the verified specification. Together with a thin, stateless REST API wrapper, this comprises the server side of CoCon. The user interface is rendered in the browser using JS.

BD security is very general; e.g., it can capture the security properties expressible in MAKS [11]. This generality copes seamlessly with complex information flow specifications, but of course comes with a price. First, allowing policies parameterized by arbitrary HOL formulas is not a priori compositional—finding correct (and manageable) compositionality conditions a la MAKS is ongoing work.

Second, generality is an impediment to automation: the CoCon confidentiality properties were verified in Isabelle in two person-months! Ongoing work focuses on streamlining automation of BD-security proofs along the following route: abstraction of an infinite system (unbounded in the number of users and documents) into a finite system, followed by model checking using HyperCTL* [12], a temporal logic able to express a large subclass of BD security. To provide a sound notion of abstraction (suitable for CoCon and for other role-based document management systems) we investigate property-specific cut-off results, which reduce the infinite-state model-checking to a finite-state verification problem with a fixed number of documents and users representing various roles: e.g., one paper, one PC member, two authors. This would allow us to decide BD-security of the original unbounded system by model-checking the system of cut-off size.

B. Client Runtime Enforcement

On the client side, we provide monitored semantics for the JS language obtained by instrumenting WebKit's JS interpreter [13], which has been extended to cover the shared state (DOM) and the event handling mechanism of the browser [14]. We have built and proved sound a formal model of the JS interpreter, the DOM and the event handling loop of the browser enhanced with checks for information flow control. Currently, the security levels are produced based on the domain from which the script is loaded and confidentiality guarantees are provided in accordance with a reactive-noninterference-like property for a termination insensitive attacker. Our monitor is implemented in a fully functional real-world browser (WebKit/Safari).

C. Server-Client Integration

As we have seen, server-side security is expressed by complex policies involving declassification under certain conditions and within certain bounds. The client side needs to preserve these conditions and bounds, that is, act as a non-leaking intermediary between the server and the end user. For instance, the verified server ensures that the content of a paper's review is only delivered to users with suitable credentials. The client-code monitor only needs to be aware of the specific sources and sinks associated to the action of outputting the review content, and make sure that nothing interferes on this route. Fortunately, this property is BD-security agnostic, and in fact can be captured by the notion of noninterference supported by our client-side monitoring tool. Hence, the JS monitor will know nothing about the specific properties enforced by the server, but will receive from the server the source-sink information in a special format along with the HTML content.

In summary, we advocate the loose integration of server-side verification with client-side monitoring: the monitor only receives from the server the information needed to preserve security. We hope our work is a first step towards a methodology for end-to-end verification of web applications.

Acknowledgment. Work funded by the DFG under the projects IFC4BC, MORES, SecDed, and SpAGAT in the priority program Reliably Secure Software Systems (SPP 1496).

References

[1] D. Guarini, "Experts Say Facebook Leak Of 6 Million Users' Data Might Be Bigger Than We Thought," in *The Huffington Post*, 2013, www.huffingtonpost.com/2013/06/27/facebook-leak-data_n_3510100.html.

[2] D. Jang, R. Jhala, S. Lerner, and H. Shacham, "An empirical study of privacy-violating information flows in JavaScript web applications," in *CCS*, 2010, pp. 270–283.

[3] J. R. Mayer and J. C. Mitchell, "Third-party web tracking: Policy and technology," in *S&P*, 2012, pp. 413–427.

[4] R. C. Phan and H. Ling, "On the insecurity of the Microsoft Research conference management tool (MSRCMT) system," in *CITA*, 2005, pp. 75–79.

[5] M. Arapinis, S. Bursuc, and M. Ryan, "Privacy supporting cloud computing: Confichair, a case study," in *POST*, 2012, pp. 89–108.

[6] S. Kanav, P. Lammich, and A. Popescu, "A Conference Management System with Verified Document Confidentiality," in *CAV*, 2014, pp. 167–183.

[7] "Reliably Secure Software Systems (RS^3)," 2014. [Online]. Available: http://www.spp-rs3.de/

[8] "The Isabelle'14 Workshop (part of the Vienna Summer of Logic)," 2014, www.easychair.org/smart-program/VSL2014/Isabelle-index.html.

[9] "The 24th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)," 2015, http://tableaux2015.ii.uni.wroc.pl/.

[10] T. Nipkow, M. Wenzel, and L. C. Paulson, *Isabelle/HOL: A Proof Assistant for Higher-order Logic*. Springer-Verlag, 2002.

[11] H. Mantel, "On the composition of secure systems," in *S&P*, 2002, pp. 88 – 101.

[12] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez, "Temporal logics for hyperproperties," in *POST*, 2014, pp. 265–284.

[13] A. Bichhawat, V. Rajani, D. Garg, and C. Hammer, "Information flow control in WebKit's JavaScript bytecode," in *POST*, 2014, pp. 159–178.

[14] V. Rajani, A. Bichhawat, D. Garg, and C. Hammer, "Information Flow Control for Event Handling and the DOM in Web Browsers," in *CSF*, 2015, To appear.